UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/782,405	02/12/2001	Clemente Izurieta	10006526-1	1881

22879 7590 09/13/2004

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

EXAMINER	
KANG, INSUN	

ART UNIT	PAPER NUMBER
2124	

DATE MAILED: 09/13/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/782,405	IZURIETA, CLEMENTE	
	Examiner	Art Unit	
	Insun Kang	2124	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE ____ MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 5/25/2004 and 7/9/2004.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1,3-7,9-11 and 20-23 is/are pending in the application.

4a) Of the above claim(s) 2, 8, and 12-19 is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1, 3-7, 9-11, and 20-23 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date 7/9/2004.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to the amendment filed 5/25/2004 and 7/9/2004.

2. As per applicant's request, claims 2, 8, and 12-19 have been cancelled, claims 1, 3-7, and 9-11 have been amended and claims 20-23 have been added. Claims 1, 3-7, 9-11, and 20-23 are pending in the application.

Specification

3. The objection to the specification has been withdrawn due to the amendment to the specification.

Claim Objections

4. The objections to claims 1 and 9 have been withdrawn due to the amendments to the claims.

Claim Rejections - 35 USC § 112

5. The applicant recites, "Applicant has removed the objectionable language in claims (Remarks, page 5)." It is noted that the claims 1, 3-6, and 9-11 were rejected under 35 U.S.C. 112, second paragraph, due to the term "substantial" being a relative term, which renders the claim indefinite. The applicant amended the claims and therefore, the scope of theses claims has changed due to the claim limitations. Accordingly, The rejections to the claims 1, 3-6, and 9-11 have been withdrawn due to the amendment to the claims.

6. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter, which the applicant regards as his invention.

7. Claims 9 and 11 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

In claims 9 and 11, it is unclear as to which "object oriented data" and "object oriented model" they are referring. They are interpreted as "the object oriented model."

Claim Rejections - 35 USC § 102

8. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

9. Claims 1, 3-6, and 9-11 are rejected under 35 U.S.C. 102(b) as being anticipated by Applicant's Admitted Prior Art (hereinafter referred to as "APA") disclosed in the instant application.

Per claim 4:

APA teaches:

-loading memory with non-object oriented data ("there are several techniques that use zero-size objects to achieve a mapping between non-object oriented

data models and object oriented based models. Zero-sized object mapping is a form of overlying existing memory occupied by another object. Some prior coding techniques to achieve zero-size object mapping are reinterpret casts," APA, page 2; "legacy infrastructures...CAD," page 1)

- mapping an object oriented model onto a memory space occupied by the non-object oriented data without requiring additional memory space ("there are several techniques that use zero-size objects to achieve a mapping between non-object oriented data models and object oriented based models," APA, page 2)

- retrieving a non-object oriented data element from the memory in the object oriented model based on the mapping ("there are several techniques that use zero-size objects to achieve a mapping between non-object oriented data models and object oriented based models. Zero-sized object mapping is a form of overlying existing memory occupied by another object. Some prior coding techniques to achieve zero-size object mapping are reinterpret casts," APA, page 2) as claimed.

Per claim 6:

Inheritance is one of important concepts in object-oriented programming (OOP). APA recites, "Another concept in object oriented programming is inheritance. Inheritance is the ability to derive a new class from one or more existing classes. The new class, known as a subclass, may inherit or incorporate all properties of a base class, including its attributes and its methods (page 4 lines 19-22)." Also,

the applicant states, "When a programmer creates an instance of that structure in memory, the programmer has to allocate memory to hold that information. In order to map, the object oriented system inherits the non-object oriented data that came from the C structure. Inheritance allows the programmer to access the non-object oriented structure or any other base structure's data (page 5 lines 9-10)." Non-object oriented data cannot be directly inherited because non-object oriented data structure does not have the concept of class mechanism. The specification does not describe how this inheritance used "to access the non-object oriented structure or any other base structure's data" and to create "child class A 210" that represents the "definition of the object oriented based object A 205 and drives it from the non-object oriented C structure 200" works differently from the conventional OOP inheritance mechanism that uses a base wrapper class to access legacy data. Therefore, the examiner considers that the inheritance mechanism stated in the instant specification (page 5 lines 7-14) is the conventional inheritance mechanism. Accordingly, APA teaches deriving a class from the non-object oriented data (page 5 lines 7-14) as claimed.

Per claim 5:

The rejection of claim 4 is incorporated, and further, this claim is another version of the claimed method discussed in claim 6 above, wherein all claim limitations also have been addressed and/or covered in cited areas as set forth the above.

Therefore, accordingly, APA teaches inheriting the non-object oriented data from memory.

Per claim 9:

The rejection of claim 4 is incorporated, and further, Inheritance is one of important concepts in object-oriented programming (OOP). APA recites, "Another concept in object oriented programming is inheritance. Inheritance is the ability to derive a new class from one or more existing classes. The new class, known as a subclass, may inherit or incorporate all properties of a base class, including its attributes and its methods (page 4 lines 19-22)." Also, the applicant states, "When a programmer creates an instance of that structure in memory, the programmer has to allocate memory to hold that information. In order to map, the object oriented system inherits the non-object oriented data that came from the C structure. Inheritance allows the programmer to access the non-object oriented structure or any other base structure's data (page 5 lines 9-10)." Non-object oriented data cannot be directly inherited because non-object oriented data structure does not have the concept of class mechanism. The specification does not describe how this inheritance used "to access the non-object oriented structure or any other base structure's data" and to create "child class A 210" that represents the "definition of the object oriented based object A 205 and drives it from the non-object oriented C structure 200" works differently from the conventional OOP inheritance mechanism that uses a base wrapper class to access legacy data. Therefore, the examiner considers that the inheritance

mechanism stated in the instant specification (page 5 lines 7-14) is the conventional inheritance mechanism. Accordingly, APA teaches accessing the non-object oriented data using an object oriented model as claimed.

Per claim 10:

The rejection of claim 4 is incorporated, and further, APA teaches retrieving occurs with zero size memory ("there are several techniques that use zero-size objects to achieve a mapping between non-object oriented data models and object oriented based models. Zero-sized object mapping is a form of overlying existing memory occupied by another object. Some prior coding techniques to achieve zero-size object mapping are reinterpret casts," APA, page 2) as claimed.

Per claim 11:

The rejection of claim 4 is incorporated, and further, this claim is another version of the claimed method discussed in claim 6 and 9 above, wherein all claim limitations also have been addressed and/or covered in cited areas as set forth the above. Therefore, accordingly, APA teaches that object oriented data inherits the non-object oriented data as claimed.

Regarding claims 1 and 3, they are the apparatus versions of claims 4 and 11, respectively, and are rejected for the same reasons set forth in connection with the rejection of claims 4 and 11 above.

Claim Rejections - 35 USC § 103

10. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

11. Claims 7 and 20-23 are rejected under 35 U.S.C. 103(a) as being unpatentable over Applicant's Admitted Prior Art (hereinafter referred to as "APA") disclosed in the instant application in view of Wise (Casting in C++: "Brining Safety and Smartness to Your Programs," Coding-zone 1996).

Per claim 7:

The rejection of claim 6 is incorporated, and further, APA discloses a reinterpret_cast but does not explicitly teach instantiating an instance of the class based on static casting. However, Wise teaches that C++ offers several different ways to cast an expression to a different type and static casting was a known programming language feature in the art of computer software development at the time applicant invention was made to "convert a base class pointer to a derived class pointer... (page 2, The static_cast Operator)" and to "perform the explicit inverse of the implicit standard conversions (page 3)." It would have been obvious for one having ordinary skill in the art of software development to modify APA's disclosed method to use static casting as the reinterpret casting result is "usually implementation dependent and, therefore, not likely to be

portable (Wise, page 7, The reinterpret_cast Operator)"and the conversion "between totally unrelated types of objects can "cause serious problems if the memory images of the types are different (APA, page 2 lines 9-14)." The modification would be obvious because one having ordinary skill in the art would be motivated to instantiate an object through static casting because it can be used to perform conversions explicitly defined in classes, to cast a pointer of a derived class to its base class implicitly, and to reverse any implicit type conversion (Wise, page 3) performing compile time type checks with certain restrictions as taught by Wise.

Per claim 20 :

APA discloses mapping object oriented data onto non-object oriented data stored in memory using zero size mapping ("there are several techniques that use zero-size objects to achieve a mapping between non-object oriented data models and object oriented based models. Zero-sized object mapping is a form of overlying existing memory occupied by another object. Some prior coding techniques to achieve zero-size object mapping are reinterpret casts," APA, page 2)

APA discloses a reinterpret_cast but does not explicitly teach static casting a non-object oriented data element from the non-object oriented data with an object oriented data element. However, Wise teaches that C++ offers several different ways to cast an expression to a different type and static casting was a known programming language feature in the art of computer software

development at the time applicant invention was made to "convert a base class pointer to a derived class pointer... (page 2, The static_cast Operator)" and to "perform the explicit inverse of the implicit standard conversions (page 3)." It would have been obvious for one having ordinary skill in the art of software development to modify APA's disclosed method to use static casting as the reinterpret casting result is "usually implementation dependent and, therefore, not likely to be portable (Wise, page 7, The reinterpret_cast Operator)"and the conversion "between totally unrelated types of objects can "cause serious problems if the memory images of the types are different (APA, page 2 lines 9-14)." The modification would be obvious because one having ordinary skill in the art would be motivated to instantiate an object through static casting because it can be used to perform conversions explicitly defined in classes, to cast a pointer of a derived class to its base class implicitly, and to reverse any implicit type conversion (Wise, page 3) performing compile time type checks with certain restrictions as taught by Wise.

Per claim 21:

The rejection of claim 20 is incorporated, and further, APA teaches retrieving the non-object oriented data element in a response to a request for the object oriented data element ("there are several techniques that use zero-size objects to achieve a mapping between non-object oriented data models and object oriented based models. Zero-sized object mapping is a form of overlying existing memory occupied by another object. Some prior coding techniques to

achieve zero-size object mapping are reinterpret casts," APA, page 2; "legacy infrastructures...CAD," page 1) as claimed.

Per claim 22:

The rejection of claim 20 is incorporated, and further, APA teaches

-all of the non-object oriented data stored in the memory is mapped with corresponding object oriented data elements when the non-object oriented data is compiled ("there are several techniques that use zero-size objects to achieve a mapping between non-object oriented data models and object oriented based models. Zero-sized object mapping is a form of overlying existing memory occupied by another object. Some prior coding techniques to achieve zero-size object mapping are reinterpret casts," APA, page 2; "legacy infrastructures...CAD," page 1) as claimed.

Per claim 23:

The rejection of claim 20 is incorporated, and further, APA teaches

-the non-object oriented data and the object oriented data is associated with very large scale integrated circuits ("The complexity of designing a VLSI chip has produced numerous CAD automation tools that help designers with synthesis and abstraction across behavioral, structural and layout domains...The information model is an abstraction of a data model...Legacy infrastructures have been developed ... Emerging CAD infrastructures use object oriented models," APA page 1) as claimed.

12. Claims 1,3-7, 9-11, and 20-22 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent 6,305,007 to Mintz in view of Wise (Casting in C++: "Brining Safety and Smartness to Your Programs," Coding-zone 1996).

Per claim 4:

Mintz teaches loading memory with non-object oriented data ("The new property pages interact with the legacy data structures via classes which descend from the base wrapper," col 4 lines 1-34).

Mintz discloses deriving a class from a base class that wraps legacy data structure, as non-object oriented data cannot be directly inherited because non-object oriented data structure does not have the concept of class mechanism. Mintz does not explicitly teach mapping an object oriented model onto a memory space occupied by the non-object oriented data without requiring additional memory space. However, Wise teaches that C++ offers several different ways to cast an expression to a different type such as reinterpret and static casting and specifically static casting was known in the art of computer software development at the time applicant invention was made to "convert a base class pointer to a derived class pointer... (page 2, The static_cast Operator)" and to "perform the explicit inverse of the implicit standard conversions (page 3)." It would have been obvious for one having ordinary skill in the art of software development to modify Mintz's disclosed method to use the static casting mechanism to access the non-object oriented data from the base class without allocating a new memory.

The modification would be obvious because one having ordinary skill in the art would be motivated to access the legacy data using the object oriented concepts such as encapsulation and abstraction overlaying the memory where the legacy data occupy by static casting performing compile time type checks with certain restrictions (Wise, page 3) as taught by Wise.

Mintz further teaches retrieving a non-object oriented data element from the memory in the object oriented model based on the mapping ("The new property pages interact with the legacy data structures via classes which descend from the base wrapper," col 4 lines 1-34; "This class is inherited from the base wrapper class (so as to be able to access the property page framework) and the policy class associated with each property that needs manipulation," col 4 lines 30-34; see also col 6 lines 22-31; col 4 lines 60-67).

Per claim 6:

The rejection of claim 5 is incorporated, and further, Mintz teaches deriving a class from the non-object oriented data ("The new property pages interact with the legacy data structures via classes which descend from the base wrapper," col 4 lines 1-34)

Per claim 5:

The rejection of claim 6 is incorporated, and further, this claim is another version of the claimed method discussed in claim 6 above, wherein all claim limitations also have been addressed and/or covered in cited areas as set forth the above.

Therefore, accordingly, Mintz teaches inheriting the non-object oriented data from memory.

Per claim 7:

The rejection of claim 6 is incorporated, and further, Mintz does not explicitly teach instantiating an instance of the class based on static casting. However, Wise teaches that C++ offers several different ways to cast an expression to a different type such as reinterpret and static casting and specifically static casting was known in the art of computer software development at the time applicant invention was made to "convert a base class pointer to a derived class pointer... (page 2, The static_cast Operator)" and to "perform the explicit inverse of the implicit standard conversions (page 3)." It would have been obvious for one having ordinary skill in the art of software development to modify Mintz's disclosed method to use static casting mechanism to access the non-object oriented data from the base class without allocating a new memory.

The modification would be obvious because one having ordinary skill in the art would be motivated to access the legacy data using the object oriented concepts such as encapsulation and abstraction overlaying the memory where the legacy data occupy by static casting performing compile time type checks with certain restrictions (Wise, page 3) as taught by Wise.

Per claim 9:

The rejection of claim 4 is incorporated, and further, Mintz teaches accessing the non-object oriented data using an object oriented model (col 3 lines 15-35; ("The new property pages interact with the legacy data structures via classes which descend from the base wrapper," col 4 lines 1-34; "This class is inherited from the base wrapper class (so as to be able to access the property page framework) and the policy class associated with each property that needs manipulation," col 4 lines 30-34; see also col 6 lines 22-31; col 4 lines 60-67).

Per claim 10:

The rejection of claim 4 is incorporated, and further, this claim is another version of the claimed method discussed in claim 6 above, wherein all claim limitations also have been addressed and/or covered in cited areas as set forth the above. Therefore, accordingly, Mintz teaches retrieving occurs with zero size memory as claimed.

Per claim 11:

The rejection of claim 4 is incorporated, and further, this claim is another version of the claimed method discussed in claim 6 above, wherein all claim limitations also have been addressed and/or covered in cited areas as set forth the above. Therefore, accordingly, Mintz teaches object oriented data inherits the non-object oriented data as claimed.

Regarding claims 1 and 3, they are the apparatus versions of claims 4 and 11, respectively, and are rejected for the same reasons set forth in connection with the rejection of claims 4 and 11 above.

Per claims 20-22, they are another method versions of claims 7 and 10, respectively, and are rejected for the same reasons set forth in connection with the rejection of claims 7 and 10 above.

13. Claim 23 is rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent 6,305,007 to Mintz, in view of Wise (Casting in C++: "Brining Safety and Smartness to Your Programs," Coding-zone 1996), and further in view of Applicant's Admitted Prior Art (hereinafter referred to as "APA") disclosed in the instant application.

Mintz and Wise do not explicitly teach the non-object oriented data and the object oriented data is associated with very large scale integrated circuits. However, most modern chips employ VLSI architectures (or ULSI). APA specifically teaches VLSI was known in the art of computer architecture, at the time applicant's invention was made, to integrate many individual functions or systems by combining numerous transistors into a single chip. It would have been obvious for one having ordinary skill in the art of computer architecture to modify the system of Mintz and Wise to incorporate the teachings of APA. The modification would be obvious because one having ordinary skill in the art would

be motivated to integrate many object oriented or non-object oriented individual functions or systems by combining numerous transistors into a single chip as indicated by APA (page 1).

Response to Arguments

12. Applicant's arguments with respect to claims 1, 3-7, 9-11, and 20-23 have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

13. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

14. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Insun Kang whose telephone number is 703-305-6465. The examiner can normally be reached on M-F 8:30-5:30.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on 703-305-9662. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

IK
8/23/2004

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100